DOCUMENT RESUME

ED 107 259 IR 001 999

AUTHOR Losee, Robert M., Jr.
TITLE A Plan for Teaching Data Processing to Library Science Students.
PUB DATE 7 Apr 75
NOTE 11p.

EDRS PRICE MF-$0.76 HC-$1.58 PLUS POSTAGE
DESCRIPTORS Computer Programs; *Computer Science Education; *Course Descriptions; Data Processing; Graduate Students; Higher Education; *Library Education; Library Schools; *Programing; *Programing Languages; Student Projects; Textbooks
IDENTIFIERS COBOL; MIX

ABSTRACT

An outline is proposed for a library school course in data processing for libraries that is different from other such courses in that it emphasizes the operations of the computer itself over the study of library computer systems. The course begins with a study of computer hardware then moves to the teaching of assembly language using the MIX language as an example. COBOL is then taught as an example of higher level programing languages. In the final segment of the course, actual library systems are studied. This course description includes a list of appropriate texts and suggested study projects. (SL)

ED107259

A PLAN FOR TEACHING DATA PROCESSING TO LIBRARY SCIENCE STUDENTS

Robert M. Losee, Jr.

April 7, 1975

U S DEPARTMENT OF HEALTH, EDUCATION & WELFARE
NATIONAL INSTITUTE OF EDUCATION
THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

IR 001 999

PREFACE

This document is an outline of a possible program for teaching a course such as "Data Processing in Libraries" to Library Science students. It has already been used with some success. The program differs from many other course outlines for "Data Processing for Library Science Students" courses in that it spends most of its' time covering how to make the computer "work" and only a few weeks studying actual library systems. My personal belief is that it is not really necessary or advantageous to go into systems in great depth in such a course—more important is the knowledge of how the computer works and what it can do with ease and what it is more difficult to do.

Since most systems are written in Higher Level Languages such as COBOL, and COBOL is "written" in an assembly language, and the assembly language depends on the computer hardware for operation, the proposed course has been planned to start out with hardware, proceed through assembly language to COBOL, and then finally to get into systems. This way, the student, at any level of computer programming, can understand exactly what the computer is doing at all levels up to the one the programmer is working at. For example, a systems programmer could weigh the merits of having all his record fields "word aligned" in a word oriented machine with much better accuracy because of the programmers knowledge of the assembly language process that is eventually going to occur to shift a field and "mask it" if it isn't word aligned.

ASSEMBLY LANGUAGE

In the assembly language portion of the course, the first choice to be made is which assembly language to teach. I chose the language MIX, which is an assembly language for no machine but which, because it is so widely known, has been implemented on many machines which now have MIX assemblers or interpreters. MIX is a language which is supposedly "powerful enough to allow brief programs to be written for most algorithms, yet simple enough so that its' operations are easily learned." (Knuth, p. 120) MIX is unusual in one respect in that it is both decimal and binary at the same time. Thus, algorithms written in it can be used either on decimal machines such as the GE 400 SERIES or on binary machines such as the majority of machines. When Martians land, they will be able to use MIX programs on their ternary machines with few problems!

MIX was originally developed by Donald Knuth of Stanford for use in a series of books he is writing entitled The Art of Computer Programming. Volume I of the series, Fundamental Algorithms, is where MIX was introduced to the world, and it was and is a very clear introduction. The text has many good homework problems, and these are rated (very fairly) according to difficulty from yes-no problems up to current research problems in Computer Science. One excellent, but discouraging, problem is #26 which is to write a card-loading routine for getting information initially into the machine. The program is interesting as it must be less than a certain length, all machines have similar programs, and it teaches the inter-relation between hardware and software.

This problem, and others like it, should not be pushed too much because they are difficult to some people (I know of three people who dropped computer courses using Knuth's text who went on to become excellent programmers) so don't push for working programs too much.

After a few weeks, when the students have begun to grasp MIX, the following program might be presented to them for work on over the course of a month or so. Two advantages of this problem are that the problem is not self-explanitory (the programmer must learn to ask for more information) and second, the problem is not completely "correct", ie. well formulated.

Part of a Circulation System

This program will take the circulation masterfile and merge it with the activities file while updating the masterfile with data on the activities file. The masterfile is the tape kept in the computer center which tells which materials are out, to whc, and when due back (as well as when checked out). The format of the masterfile's records is:

```
12345678901234567890123456789012345678901234567890
Call no.----------------------Date-Date-Soc Sec.--
                               Out  Due  Number
```

Date Format
1 2 3 4 5
M M D D Y (Assumes Year is in 1970's)

The file is sorted so that the lowest call numbered record is the first record on the tape and the highest numbered record is the last record on the tape.

The activities file is the tape made at the circulation desk. It contains records, unsorted, of the same format as the masterfile except that instead of the "Date Out" is a set of possible entries indicating what type of activity the record is for, ie. 00001 = Discharge, 00002 = 2 week charge, 00007 = Semester charge, etc. The Date Due field will contain nothing.

Flowchart, then write a program which can do all this. If you think the problem itself must be changed because of an impossibility in the problem, you have the right to change it. NOTE: This operating system has a "canned" sort program in it. The program takes an unsorted tape (rewound) and sorts it,

producing a sorted tape (not rewound).

rI1 = number of words in sort key (words to be used in sorting)

rI2 = which word of record is first of sort key
(the first word is word 0)

rI3 = tape unit input tape is on (assume only one tape needed)

rI4 = output tape unit number

rI5 = number of words in entire record

rI6 = maximum number of records to be processed (in binary)

To use sort program, necessary information must be put in indexes; then a jump is made to "SORTb". The sort program will re-enter your program by doing a jump to "RTURN".

NOTE: The operating system, when initially starting your program, will put today's date in rI6 in the Date Format shown before. The program may use rI6 for whatever it wants during the course of execution. However, if the date is not stored somewhere before something else is put into rI6, the date is gone.

COBOL

COBOL can be taught rather easily to those who already have a good grasp of the powers of the computer such as would have been obtained by those who learned an assembly language such as MIX. There are three excellent (although dated) texts on COBOL which I chose between. The first was Saxon's COBOL; A Self-Instructional Manual (1963) which has the advantage that the students can work at their own speed and the disadvantage that some statements which are, in the real world, infrequently used are given about the same emphasis as commonly used statements and students frequently get "hung up" on some of the unused ones. Another good text is McCracken and Garbassi's A Guide to COBOL Programming, second edition, (1970) which is so comprehensive and gets so deep that, at the teachers discretion, it might be used as a text for a more advanced course or for outside reading for someone who already has some knowledge of COBOL and needs practice to develop commercial proficiency. The third book, and the one I chose, is Raun's An Introduction to COBOL Computer Programming for Accounting and Business Analysis (1966). This book was chosen for a very selfish reason—many of the examples in the book are from a GE-400 SERIES COBOL Compiler which I am intimately familiar with and found much easier to explain than the IBM 360 Compiler for COBOL 61. For a person familiar with knowledge of the 360 Compiler, and with more than five weeks to teach COBOL in, it might be best to use the McCracken and Garbassi text.

For problems in this COBOL section of the course, the teacher has the world of library automation at their feet. One note of caution is that most Library Science students have at one time or another worked at a circulation desk of some sorts while not as many have worked in a serials processing section which utilized claiming and other procedures often used in larger libraries that automate; thus it might be best to select most problem material from the area of circulation. One excellent problem would be to have the students attempt to program in COBOL the problem which they attempted to do in MIX—this is a quick way to see one of the reasons why most programs are written in Higher Level Languages. One good requirement for student programmers is that they must have <u>at least one comment line</u> in the procedure division <u>for each line of program</u> in the procedure division. This forces the student to develop good documentation habits: something often not developed until after months or years of professional programming involving modifying other people's non-documented programs.

SYSTEMS

The final segment of the course deals with actual operational systems. Two approaches may be taken here; one dealing with the actual development by the students of operational (on paper or on the computer) systems and the other approach being the study of pre-existant systems. A mixture of these two is quite possible and might be the most profitable course. For texts, one might consider using Case Studies in Library Computer Systems by Palmer and/or almost any articles in the Journal of Library Automation. Studying one or two of the LARC reports in depth immediately after leaving COBOL has the advantage that an actual system (ie. circulation) can be viewed at great depth through systems flowcharts and the actual examples of inputs and outputs which are printed by LARC

BIBLIOGRAPHY

Knuth, Donald E. Fundamental Algorithms. Reading, Mass., Addison-Wesley, 1968.

LARC Reports Library Automation At The University of Wisconsin-Milwaukee volume 6/issue 3, 1973.

McCracken, D.D. and Garbassi, Umberto A Guide to COBOL Programming, second edition, New York, Wiley-Interscience, 1970.

Palmer, Richard P. Case Studies in Library Computer Systems. New York, Bowker, 1973.

Ruan, Donald L. An Introduction to COBOL Computer Programming for Accounting and Business Analysis. Belmont, California, Dickenson, 1966.

Saxon, James A. COBOL; A Self-Instructional Manual. Englewood Cliffs, N.J., Prentice-Hall, 1963.